Poster: LockedApart: Faster GPU Fingerprinting Through the Compute API

Tomer Laor and Yossi Oren

Ben-Gurion Univ. of the Negev, Israel
`tomerlao@post.bgu.ac.il`
`yos@bgu.ac.il`

Abstract. WebGL offers website direct access to the GPU, allowing beautiful graphics. The direct hardware access offered by WebGL was also shown to expose multiple security vulnerabilities. In particular, DrawnApart [2] showed that by performing graphical micro-benchmarks on the GPU, it is possible to fingerprint the underlying hardware. Recently, the access of websites to the GPU was extended with the introduction of the WebGPU API, a new low-level API that allows websites to perform general-purpose computations on the GPU. Our research question was: *Does this additional access to the GPU expose additional avenues for fingerprinting?* Our initial results show that this is true. Our new attack, which we call LockedApart , uses WebGPU to directly measure contention between GPU threads. Compared to DrawnApart , LockedApart is up to 310x faster and up to 1.8x more accurate. These preliminary results show that it is important to consider the security implications of this new API. The code for LockedApart is available at `https://github.com/LockedApart/LockedApart`.

1 Introduction

General-purpose computing on GPUs (GPGPU) has revolutionized various fields, enabling high-performance computations across domains such as machine learning, scientific simulations, and cryptography. WebGPU is a next-generation compute-focused API designed to replace WebGL, which was primarily geared toward graphical rendering. Unlike WebGL, WebGPU provides access to low-level GPU features, such as shared memory across threads. These features open new opportunities for high-performance computing within browsers, transforming the web into a powerful platform for general-purpose computation.

Device fingerprinting is a technique used to identify and distinguish devices based on their unique characteristics. The most common device fingerprinting techniques are deterministic, such as installed fonts, screen resolution, browser plugins, etc. These techniques are unable to track a single device for an extended duration due to the constant changes in the device's configuration, which causes the fingerprint to evolve and be confused with other fingerprints [3]. Microarchitectural fingerprinting is a technique that uses the unique characteristics of a device's underlying hardware to identify and distinguish devices, which results enhances the tracking duration of devices [2]. The current state-of-the-art microarchitectural fingerprinting technique, DrawnApart [2], leverages the WebGL API to collect microarchitectural fingerprints of devices. DrawnApart uses contention between GPU execution units to create a fingerprint. To measure contention, DrawnApart measures the time it takes to render a canvas element that is stalled by a computation that runs concurrently on multiple execution units of the GPU.

In this work, we leverage the new WebGPU API to collect microarchitectural fingerprints of devices remotely. This new API allows us to move away from graphical rendering and use compute features, such as shared memory across GPU threads. This allows us to collect fingerprints faster since we can measure contention between GPU threads directly, without the need to render a canvas element and time the rendering process.

2 Method

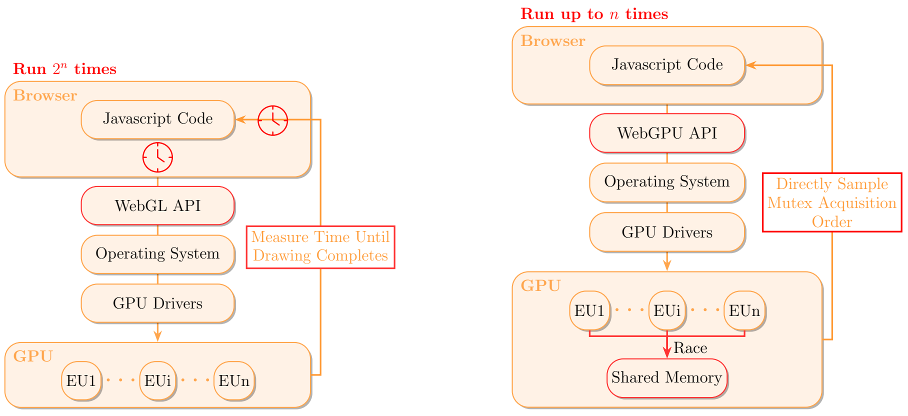
fig. 1 illustrates our method and compares it to DrawnApart . Our method spins up n GPU threads that run concurrently and share a mutex. n should be lower than the number of execution of the GPU. Each thread runs a loop that tries to acquire the mutex. If a thread acquires the mutex, it increments a shared counter and writes the acquisition index to a shared array in the location of the thread ID. If a thread fails to acquire the mutex since it's already taken, it continues to the next iteration of the loop. The number of iterations is the number of threads that we spin on the GPU. In a case that the number of execution units inside the GPU is small, for example, in integrated GPUs, to have a bigger trace we can run our method k times one after the other and concat the traces. We input the resulting trace into a random forest classifier to classify the device. Compared to DrawnApart , our method does not require drawing or using timers.

3 Preliminary Results

We evaluated LockedApart and DrawnApart on:

1. Six identical machines featuring Intel i5-10500 CPUs and Nvidia GTX 1650 GPUs, running Windows 10 (build 19045.5247), Chrome (version 131.0.6778.86), and Brave (version $1.71.123 - 93$).
2. Four identical Samsung Galaxy S23 phones from the Samsung Remote Test Lab [1], running Android 14 and Chrome (version 132.0.6834.165).

For the machines equipped with Nvidia GTX 1650 GPUs, we used $n = 256$ threads and $k = 1$ repetitions. For the Galaxy S23 phones, we used $n = 10$ threads and $k = 10$ repetitions. For each evaluation, we collected 750 traces and applied 5-fold cross-validation to split the data into training and testing sets. We



(a) (1) DrawnApart method. (b) (2) LockedApart method.

Fig. 1: Comparison between LockedApart and DrawnApart . Changes highlighted in red.

were able to collect only 62 DrawnApart trace on Galaxy S23 phones, due to its very slow collection time - around 16 seconds per trace and Samsung Remote Test Lab's session time limit. A Random Forest classifier with 600 estimators was used, keeping the other parameters as sklearn's default. As shown in fig. 2, on machines equipped with the Nvidia GTX 1650 GPU, LockedApart is 16 times faster and delivers 1.5 times higher accuracy on Chrome, and 1.2 times higher accuracy on Brave, compared to DrawnApart . On the Samsung Galaxy S23, LockedApart is 310 times faster and achieves 1.8 times higher accuracy on Chrome compared to DrawnApart .

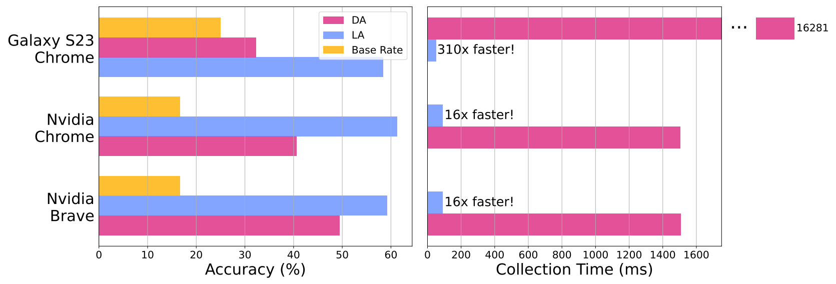


Fig. 2: Shows the accuracy and collection time of LockedApart and DrawnApart .

4 Conclusion & Future Work

In this work, we introduce LockedApart , a novel GPU fingerprinting technique that leverages the WebGPU API to achieve faster and more accurate microarchitectural fingerprints compared to DrawnApart . By measuring contention between GPU threads directly, LockedApart demonstrates a significant improvement in both speed and accuracy across different browsers.

Future work will focus on extending our evaluation to clusters with more identical machines. Additionally, we plan to test LockedApart on a broader range of GPU models and vendors to generalize its applicability and robustness. Finally, we aim to investigate how the passage of time since data collection impacts the performance of LockedApart .

References

1. Samsung Remote Test Lab. `https://developer.samsung.com/remotetestlab`
2. Laor, T., Mehanna, N., Durey, A., Dyadyuk, V., Laperdrix, P., Maurice, C., Oren, Y., Rouvoy, R., Rudametkin, W., Yarom, Y.: DRAWN APART: A device identification technique based on remote GPU fingerprinting. In: 29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022. The Internet Society (2022), `https://www.ndss-symposium.org/ndss-paper/auto-draft-242/`
3. Vastel, A., Laperdrix, P., Rudametkin, W., Rouvoy, R.: FP-STALKER: tracking browser fingerprint evolutions. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 728–741. IEEE Computer Society (2018). `https://doi.org/10.1109/SP.2018.00008`, `https://doi.org/10.1109/SP.2018.00008`